Exploring i-Vector Based Speaker Age Estimation

Anna Silnova

Master's Thesis

ITÄ-SUOMEN YLIOPISTO

Faculty of Science and Forestry

School of Computing

June 2015

UNIVERSITY OF EASTERN FINLAND, Faculty of Science and Forestry, Joensuu
School of Computing

Student, Anna Silnova : Exploring i-Vector Based Speaker Age Estimation
Master's Thesis , 42 p.
Supervisors of the Master's Thesis : PhD Tomi Kinnunen
June 2015

Abstract:

We address the problem of speaker age estimation. Inspired by recent positive results in this task using i-vectors, we further analyze their application to age estimation. We cover various aspects of the age estimation process. First, we consider front-end processing, including different i-vector extraction. We analyze effects of network architecture, training algorithm and ensembles of several networks. Our experimental findings on the NIST 2008 and 2010 SRE corpora indicate that, among all the studied setups, ANN back-end in combination with i-vectors worked the best. This approach lead to mean absolute errors (MAEs) of 5.49 years for females and 6.35 years for males. These correspond to 4.5% relative improvement in comparison to our state-of-the-art baseline for both females and males.Thus, the choice of back-end was not found to affect the final prediction accuracy much compared to choice of features used for age estimation. Consecuently, for the future work, we propose to concentrate attention on new types of acoustic features rather than on back-end methods.

Keywords: artificial neural networks, automatic age estimation, i-vectors, multilayer perceptron
CR Categories (ACM Computing Classification System, 1998 version): A.m, K.3.2

Foreword

The author would like to thank Tomi Kinnunen (University of Eastern Finland), Ondrej Glembek and Pavel Matejka (both from Brno University of Technology) for supporting and advising her during the whole work process from experiments to writing the thesis text. Also, great thanks to M.H.Bahari for sharing trial lists for age estimation used in his works.

The work was partially carried as ERASMUS exchange of the author, during her visit to Brno University of Technology.

List of abbreviations

ANN	Artificial neural network
BFGS	Broyden–Fletcher–Goldfarb–Shanno algorithm
CMVN	Cepstral mean and variance normalization
DCT	Discrete cosine transform
DFT	Discrete Fourier transform
GMM	Gaussian mixture model
JFA	Joint factor analysis
LDA	Linear discriminant analysis
MAE	Mean absolute error
MFCC	Mel-frequency cepstral coefficient
MLP	Multilayer perceptron
SDC	Shifted delta cepstral coefficient
SGD	Stochastic gradient descent
SRE	Speaker recognition evaluation
SVM	Support vector machine
SVR	Support vector regression
UBM	Universal background model
WCCN	Within-class covariance normalization

List of symbols

S^{tr}	Training set for age estimation
X_p	p^{th} speech utterance in the training set ($1 \leq p \leq P$)
Y_p	p^{th} age label corresponding to X_p ($1 \leq p \leq P$)
X^{test}	Unseen test utterance
Y^{test}	Predicted age label for X^{test}
$h(n)$	Window function in time domain
$X(k)$	k^{th} component of DFT of signal x ($0 \leq k < N$)
$\text{Mel}(f)$	Mel-frequency corresponding to "physical" frequency f
ν_m	Output of m^{th} channel of mel filter bank ($1 \leq m \leq M$)
C_t	t^{th} cepstral coefficient for a given frame ($0 \leq t \leq T$)
ΔC_t^i	Delta coefficient for frame i
N_{cep}	The number of base MFCCs
E	Logarithm of time-domain signal energy
l	The number of frames in a speech utterance
d	Dimensionality of a feature vector
$p(u\|H_i)$	Likelihood of the hypothesis that an utterance u was generated by speaker S_i
$w_j, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j$	Parameters (weight, mean vector and covariance matrix) for Gaussian j in GMM
K	The number of Gaussian components in GMM
$n_j, E_j(x), E_j(x^2)$	Probabilistic count, first- and second-order moments used to compute parameters of the j^{th} mixture component in MAP adaptation
Ξ	The number of frames used for adaptation
$\hat{w}_j, \hat{\boldsymbol{\mu}}_j, \hat{\boldsymbol{\sigma}}_j$	Adapted mixture parameters for the j^{th} Gaussian
$\alpha_j^w, \alpha_j^\mu, \alpha_j^\sigma$	Adaptation coefficients used for calculating $\hat{w}_j, \hat{\mu}_j, \hat{\sigma}_j$
r	Relevance factor in MAP adaptation
$\boldsymbol{M}$	GMM supervector
$\boldsymbol{s}, \boldsymbol{c}$	Speaker and channel supervectors
$\boldsymbol{m}$	UBM supervector
$\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{D}$	Hyperparameters of JFA (eigenchannel, eigenvoice and residual matrix)
$\boldsymbol{T}$	Total variability matrix (i-vector extractor)
$\boldsymbol{v}$	I-vector
$\boldsymbol{B}$	WCCN transformation matrix
$\boldsymbol{S}_b, \boldsymbol{S}_w$	Between- and whithin-class scatters used in LDA
$\boldsymbol{a}, b$	Input and output of SVM,SVR and ANN models

$\boldsymbol{\eta}$	Weight vector in SVM, SVR and ANN
$f(\boldsymbol{a})$	Prediction function in SVM and SVR
$\Phi(\boldsymbol{a})$	Mapping function to high-dimensional space in Kernel classifiers
$K(\boldsymbol{a}, \boldsymbol{a}_n)$	Kernel function
α_n	Lagrange multipliers used for optimization of objective function in SVM and SVR
ϵ	Acceptable distance for training points to lay from SVR prediction function
C	Regularization parameter in SVR
H	Approximation of Hessian matrix used in the BFGS training algorithm
α	Step size for training algorithm of ANN
λ	Regularization parameter in ℓ_2 regularization
ρ	Pearson's correlation coefficient
$\theta, \chi, \delta, \kappa$	Parameters user to calculate SDC features
$s_{i\theta+j}$	SDC feature

Contents

1 Introduction

In everyday communication, humans use not only semantic (what is said) but also *paralinguistic* information. The latter one is related to speaker characteristics, such as her gender and age. Recognition of human age based on voice is feasible for a human listener. A listener may not be accurate in detecting the precise age, but may guess the broad age group. But for computer to recognize age of the speaker is a challenging task. To solve the task, a lot of speech data with age labels is needed.

Nowadays, the internet provides a wide range of possibilities for services. In the context when there is no direct contact with the client, information about him such as language, gender or age can be helpful for delivering or recommending appropriate products and services [1, 2]. Human voice is a great source of information, and, since popularity of vocal user-computer interaction has considerably increased within the past decades, possibility to use voice recordings of the user for retrieving meta-information becomes easier. To exemplify, some studies were conducted to implement age recognition system built in the TV set in order to provide targeted advertisements for the people watching it [1]. Other examples of commercial applications of age recognition include smart homes or car systems which can adapt to the needs of the targeted user.

Automatic age recognition system can be useful tool in forensic applications, too [3]: it could help to narrow down a list of suspects when a speech sample is used as an evidence. Such a situation could take place in the cases of blackmailing calls, for instance. Another example of automatic age recognition system is an Android mobile application, *Reco Lab*, developed by the author of this thesis in cooperation with her fellow student Ivan Kukanov. This application is aimed at recognizing the speaker's identity, age and gender. Fig. 1 presents several screenshots of this program. Baseline method of this thesis is used for age recognition in this project.

The problem of automatic speaker *age recognition* can be formally cast as follows: we are given a training set of P speech recordings: $S^{\text{tr}} = (X_1, Y_1), ..., (X_P, Y_P)$, where X_p and Y_p are, respectively, the p^{th} speech utterance and its age label. The goal is to create a system, which will predict, for an unseen utterance X^{test}, its label Y^{test} accurately.

Age recognition can be approached in two different ways. The first way is to consider it as a *classification* task. This means that all the range of possible ages is divided into several bins (e.g. 5-years wide), and then for the target utterance decision Y^{test} is the index of the bin where this utterance will be placed or the set of probabilities for the utterance to be a member of each bin. The second approach is to consider the task as a *regression* problem: for each utterance, predict not the age group but the actual age. In this case, Y^{test} is real-valued rather than categorical. The first approach was widely studied in the past decade but it has some obvious drawbacks: for example, two persons with age difference of just one year could be placed into two different

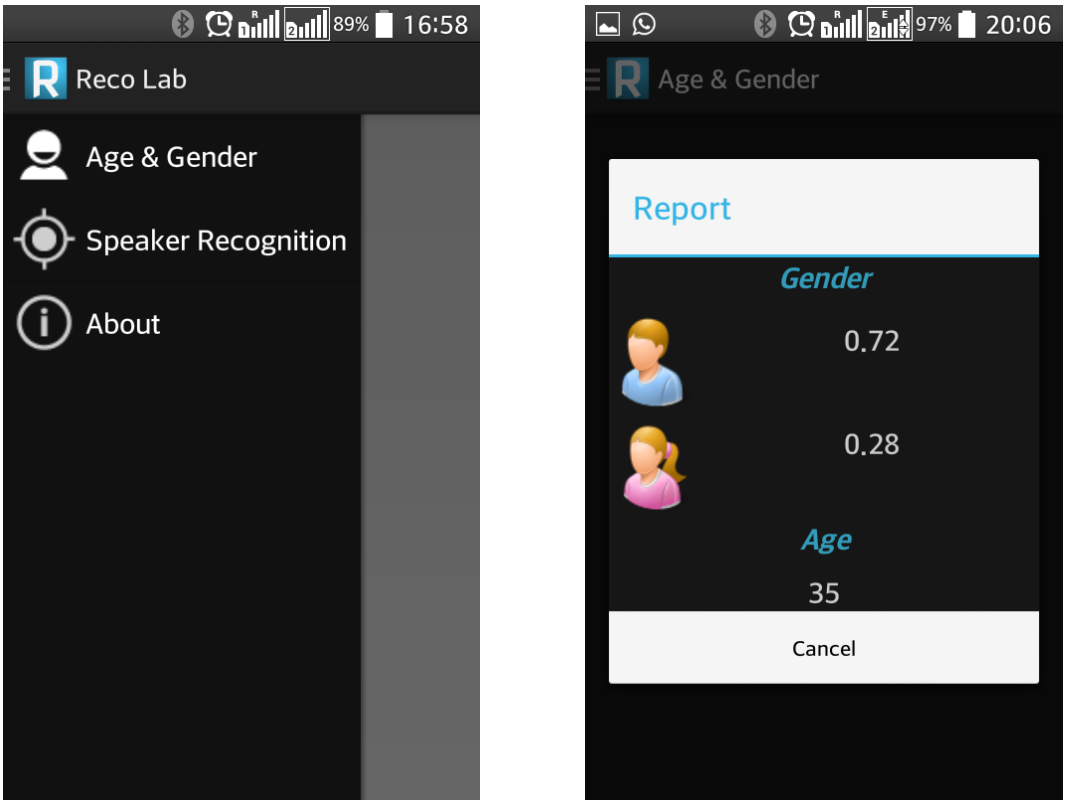

Figure 1: Reco Lab mobile application. Main menu and report of the Age&Gender recognition module.

age groups, but at the same time people with much greater age difference can be members of the same class. Hence, regression seems more natural for recognizing continuous values such as age. This thesis concentrates on regression rather than classification view of age estimation.

One of the most successful approaches for age estimation is based on so-called *i-vectors*. I-vector gives a representation of speech utterance in the form of low-dimensional feature vector. Originally introduced for the task of speaker verification [4], i-vectors have helped to increase classification accuracy and simplify classification of variable-duration utterances. Afterwards, i-vectors have also been applied to language [5] and accent [6] recognition. It seemed therefore logical to apply the same technique for age estimation, which was successfully done recently in [7, 8]. In those studies, prediction of the target speaker age was done by *support vector regression* (SVR) — one technique for function approximation. The general idea of using SVR for age estimation was introduced previously in [9], based on *GMM supervectors* [10], which is another way of one-vector speech utterance representation (concepts of i-vectors, GMM supervectors and SVR will be described in detail in subsequent sections). Considerable improvement of the age recognition accuracy was made by simple replacement of supervectors by i-vectors. The main research direction in this thesis is further development of i-vector approach in age estimation task. Particularly, we consider various setups for i-vector extraction.

Artificial neural network (ANN) [11] is an approach originally built in attempts to find mathematical representation of neural processes of the human brain. From a practical point of view, high accuracy in representing brain information processing is not necessary; simpler models can produce good results in practical tasks. Neural networks are frequently used in different pattern recognition and machine learning tasks, and proved to be efficient. Thus, it seems reasonable to study this approach in age estimation, with the hope that it can outperform previously used methods. Thus, a core part of this thesis is to explore various neural network architectures for the task of age recognition. Our prime hypothesis is that applying neural network for age recognition can help outperform previous approaches utilizing SVRs. To this end, we analyze effects of network architecture, training algorithm and ensembles of several networks.

The remainder of the thesis is organized as follows. Section 2 describes briefly the earlier key literature in age estimation. Section 3 is intended as a self-contained description of the methods used in the experimental part. It is targeted for a reader who is familiar with basics of signal processing and pattern recognition but less familiar with state-of-the-art speaker recognition machinery. The next three Sections – 4, 5 and 6 – are then focused on age estimation methods and experimentation. To ensure that our findings are comparable to results reported earlier in literature, we have adopted the exact same set-up as used in two recent studies [7, 8]. We first replicate the baseline method (Section 4) of [7, 8] from scratch to increase confidence that it is correctly implemented and represents state-of-the-art in age estimation. Sections 5 and 6 then seek to improve the baseline following several modifications described above. Finally, Section 7 summarizes our findings and gives recommendations how to configure ANN back-end for age estimation.

2 Previous work

Although studies on influence of speaker age on speech characteristics have been conducted since 1950s [12, 13], actual systems attempting to estimate age from human voice appeared only in early 2000s. Partly, it may have been caused by lack of appropriate datasets. During the past decade, such corpora were collected, including, for instance, "aGender" [14], NIST SRE 2008 and NIST SRE 2010 datasets. Since then many different age estimation approaches were studied. Many of these methods were adopted from the field of speaker recognition.

One of the first automatic age recognition systems was studied by Minematsu *et. al* [15]. In their study, the authors considered it as a classification task with only two classes: elderly people and all the rest. Having 43 speakers for each class, the authors trained Gaussian mixture model based on acoustic features. With this approach, a correct identification rate of 91% was reported. In later studies it was proposed to divide speakers into more age groups and combine age recognition together with gender recognition, e.g. in [16] the task was to classify speakers into 8 groups depending on their age and gender. The authors used high level features, such as pitch, number of speech pauses and duration of pauses among others. They studied several techniques for classification including naive Bayes, decision trees and support vector machines. The best performance was achieved with artificial neural networks, leading to 65.5% accuracy for 8-class problem and 94.6% accuracy for classification of elderly speakers and others (the same task as in [15]).

In 2010, age sub-challenge [17] of *Interspeech paralinguistic challenge* was presented. In this challenge, all speakers from the "aGender" corpus were divided into 4 age groups. "aGender" contains around 70,000 utterances from 772 speakers, all utterances being telephone recordings. To enhance accuracy of age recognition were suggested systems using GMM supervectors [18], fuzzy SVM modeling [19] and Maximum-mutual information (MMI) with GMM and SVM [20]. The latter work became the winner of the challenge, the classification accuracy being 55.3%. A brief overview of the systems applied for age sub-challenge is given in [3], which also proposed a way to combine acoustic and prosodic information yielding 52.2% classification accuracy for the same experimental setup; that is, the fusion was unsuccesful.

Due to drawbacks of classification approach to age recognition, later it was suggested to consider it as a regression task. Starting with [9], support vector regression (SVR) systems were developed in combination with different types of features. Firstly, GMM supervectors were used for age estimation in [9], involving the task of age estimation of children from preschool and primary school. In total, 212 children were selected for the experiment, each having 99 utterances and age of children varying from 5 to 11 years old. As performance measures, Pearson's correlation and Spearman's correlation, which is Pearson's correlation between two ranked variables, coefficients were used, achieving values 0.89 and 0.83, respectively. The mean deviation

of predicted age was 10.3 months. Another study on SVR and GMM supervectors applied for age estimation [21] studied the effect of dimensionality reduction on age prediction accuracy. The authors reported that the proposed dimensionality reduction technique lead to 10% relative improvement compared to the system using full-size GMM supervectors. Finally, in [7] authors proposed using i-vectors in place of GMM supervectors, which helped to decrease mean absolute error (MAE) from 8.24 to 7.95 years for telephone recordings of female speakers in NIST 2008 and 2010 speaker recognition evaluation (SRE) datasets. This result was further improved in [8] by implementing session compensation. In that paper, the authors used two performance measures: MAE as before and Pearson's correlation, the reported results being 5.78 years and 0.80, respectively.

3 Theoretical basis

This Section introduces a set of mathematical tools that are needed in the following sections. We review the commonly used feature extraction techniques in the area of speaker recognition, which are also used for age estimation in this thesis. We further describe *support vector regression* (SVR) and *artificial neural network* (ANN) .

3.1 Short-term spectral features

The first step of any speech or speaker recognition task is to extract *features* from the given raw audio signals. Features should emphasize the relevant properties of the speech signal for a given classification task and suppress distractive information, such as background noise. There are a multitude of such techniques available, but the most popular one uses short-term power spectrum to compute so-called *mel-frequency cepstral coefficients* (MFCCs). MFCC extraction is straightforward and the features have proven effective in many applications [22]. MFCCs remain widely used even if they were introduced by Davis and Mermelstein already in the 1980's [23].

One of the main characteristics of speech signal is that it is highly nonstationary. But we can assume that in sufficiently short time intervals the signal remains stationary (i.e. its frequency content is constant). Consequently, for the processing of the speech signal, we usually divide it into short *frames* (usually 20-30 ms long) and extract a spectral feature vector for each frame. Illustration of MFCC extraction is shown in Fig. 2.

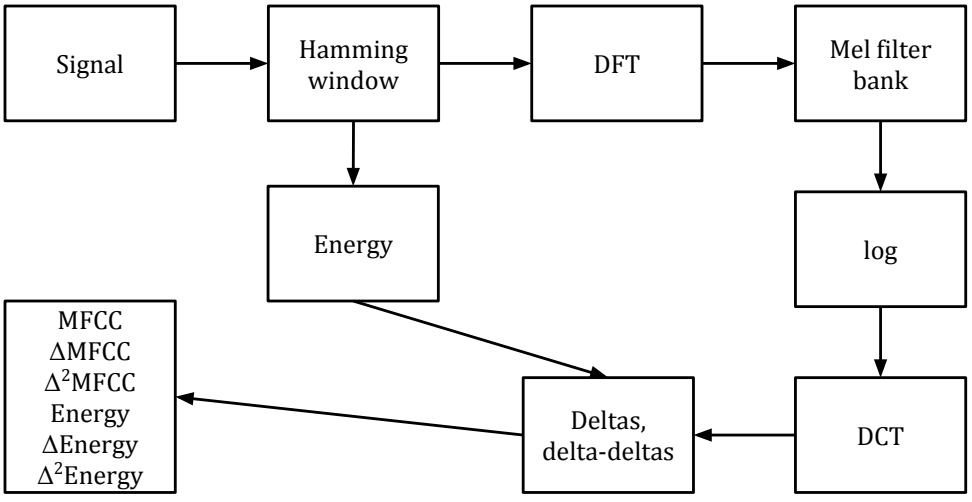

Figure 2: MFCC extraction workflow.

To extract spectral features, for each frame, *discrete Fourier transform* (DFT) is first com-

puted [24].

$$X(k) = \sum_{n=0}^{N-1} x(n)h(n)e^{\frac{-j2\pi kn}{N}}, 0 \leq k < N \tag{1}$$

DFT decomposes any signal into its frequency components. In (1), $x(n)$ denotes the signal at discrete time index n and $h(n)$ is a *window function* (usually Hamming window). The window function is used to minimize discontinuities at the begining and end of each frame. Finally, $\frac{2\pi k}{N}$ denotes the discretized frequency samples. Note that $X(k)$ is complex-valued, having both phase and magnitude information. Usually, the phase is discarded and only the magnitude (or squared magnitude) is used. They are defined as follows.

$$|X(k)| = \sqrt{\mathrm{Re}(X(k))^2 + \mathrm{Im}(X(k))^2}, \quad \arg(X(k)) = \tan^{-1}\left(\frac{\mathrm{Im}(X(k))}{\mathrm{Re}(X(k))}\right) \tag{2}$$

The next step is to use a set of band-pass filters to integrate spectral energy over the neighbouring frequencies. The common way is to use *mel-frequency* filter bank (Fig. 3), where the filters for the lower frequency range are narrower and more densely spaced than for the higher frequency range. This enables higher precision for representing the lower frequency range, based on the observation that human perception of the frequency contents of the sound is more precise for lower frequencies. Thus, the purpose of this filter is to emulate the processing steps of the human auditory system. For any given "physical" frequency f, in Hertz, the corresponding mel-frequency can be approximately calculated from following equation:

$$\mathrm{Mel}(f) = 1125 \ln(1 + f/700)$$

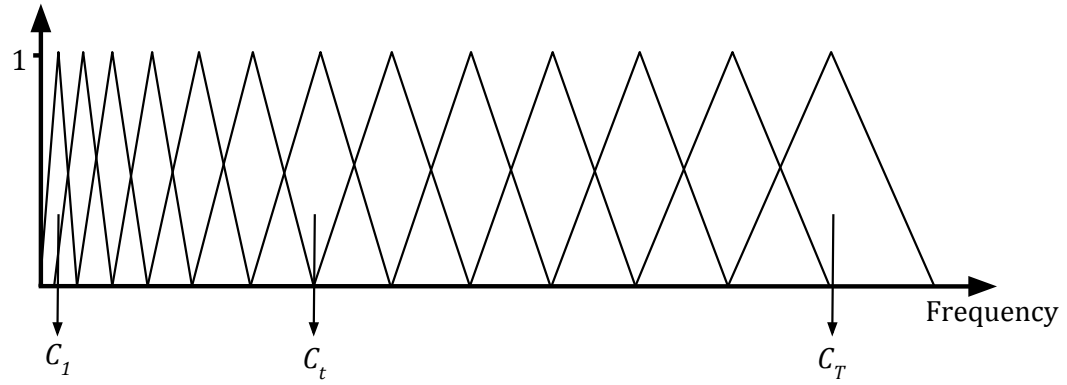

Figure 3: Mel filter bank.

Following logarithmic compression, discrete cosine transformation (DCT) is then applied

to obtain MFCCs:

$$C_t = \sum_{m=1}^{M} \log(\nu_m) \cos\left(\frac{\pi t}{M}(m - \frac{1}{2})\right) \tag{3}$$

Here, C_t is the t-th cepstral coefficient ($0 \leq t \leq T$) and ν_m is the output of the m-th channel of the mel filter bank ($1 \leq m \leq M$). Typically, only the first few cepstral coefficients are retained. The common practice is to leave only 12-20 of them. This is done because higher coefficients represent fast changes in filter bank energies, and these fast changes can degrade performance of final recognition system. DCT is applied for decorrelating the mel subband energies.

The log of the time-domain signal energy is often also added to the vector. To improve the performance of speech or speaker recognition systems one can add the first and second order time derivative estimates (also called *delta* and *delta-delta* coefficients) of the base coefficients. They can be obtained from the following formula.

$$\Delta C_t^i = \frac{\sum_{q=1}^{Q} q(C_t^{i+q} - C_t^{i-q})}{2\sum_{q=1}^{Q} q^2} \tag{4}$$

Here, ΔC_t^i is delta coefficient for frame i calculated over C_t^{i-Q} to C_t^{i+Q}. Typical value for Q is 2. Second order derivatives can be calculated in the same way, but instead of static base coefficients we have to use deltas. The final feature vector is the result of concatenating the base MFCCs with their first and second order derivatives. In this case, the final feature vector of dimentionality $3N_{\text{cep}} + 3$, where N_{cep} denotes the chosen number of base MFCCs, for a single frame takes the following form:

C_1	...	$C_{N_{\text{cep}}}$	E	ΔC_1	...	$\Delta C_{N_{\text{cep}}}$	ΔE	$\Delta^2 C_1$	...	$\Delta^2 C_{N_{\text{cep}}}$	$\Delta^2 E$

To compute the features for the whole utterance, we stack all the short-time features to matrix of size $l \times d$, where l is the number of frames in the utterance and d is the chosen dimensionality of features ($3N_{\text{cep}}+3$ in example above). For utterances of different durations, the size of the feature matrix will be different. In the following subsections, we discuss methods that enable representing varying duration utterances using fixed-dimentional utterance representation.

3.2 Gaussian mixture model and universal background model

In classification tasks, such as speaker recognition, every class is represented by its *model*. Then the decision for a particular test utterance u (represented by a sequence of acoustic features such as MFCCs) is often based on the likelihood $p(u|H_i)$ of the hypotheses that the utterance was generated by speaker S_i. Here, $p(u|H_i)$ denotes the likelihood of hypothesis H_i given a speech segment u. Utterance is classified to the class that yields the largest likelihood. An

important question, then, is how to model these likelihoods. The most common practice in speaker recognition is to model each class using a *Gaussian mixture model* (GMM) [25], as follows:

$$\begin{aligned} p(\boldsymbol{x}) &= \sum_{j=1}^{K} w_j p_j(\boldsymbol{x}) \\ p_j(\boldsymbol{x}) &= \frac{1}{(2\pi)^{\frac{d}{2}}|\boldsymbol{\Sigma}_j|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_j)'\boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_j)\right) \end{aligned} \tag{5}$$

The probability density function of a GMM is a weighted sum of K multivariate Gaussian densities where K is a parameter that one can change depending on the problem. Each Gaussian is described by its d-dimensional *mean vector* $\boldsymbol{\mu}_j$ (d is the dimensionality of feature vectors) and $d \times d$ *covariance matrix* $\boldsymbol{\Sigma}_j$. Mean of Gaussian indicates its location in the space while covariance is responsible for the orientation of the Gaussian in respect to coordinate axes. In general, $\boldsymbol{\Sigma}_j$ is a full covariance matrix, but it is common to assume that it has nonzero elements only on its diagonal, which helps decreasing the computational load, since there is much less parameters to optimize. Parameters w_i are the *prior probabilities* of the corresponding Gaussians and are constrained as: $\sum_{j=1}^{K} w_j = 1$ and $w_j \geq 0$. The parameters of the model, $w_j, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j$, can be optimized to maximize likelihood of the training data. This is usually done by *expectation maximization* (EM) algorithm, whose detailed description one can find in [26]. For more practical details an interested reader is advised to look into [11].

Based on the GMM approach, Reynolds and others introduced new ideology for the speaker verification task. The idea is based on the fact that there are only two classes in this problem: *target* speaker and *all the other possible speakers*. To model the universe of all possible speakers, so-called *universal background model* (UBM) was introduced in [25]. UBM is nothing but a GMM trained on large amount of data originating from different speakers and is intended to represent speaker-independent distribution of spectral features. The UBM can then be adapted using *maximum a posteriori* (MAP) criterion to model any target speaker (Fig. 4). This is done by first computing the following statistics:

$$\begin{aligned} n_j &= \sum_{\xi=1}^{\Xi} P(j|x_\xi) \\ E_j(x) &= \frac{1}{n_j}\sum_{\xi=1}^{\Xi} P(j|x_\xi)x_\xi \\ E_j(x^2) &= \frac{1}{n_j}\sum_{\xi=1}^{\Xi} P(j|x_\xi)x_\xi^2 \text{ , where} \\ P(j|x_\xi) &= \frac{w_j p_j(x_\xi)}{\sum_{i=1}^{K} w_i p_i(x_\xi)} \end{aligned} \tag{6}$$

These statistics are the probabilistic count and the first- and second-order moments that are used to compute mixture weight, mean and variance. Then the adapted parameters for Gaussian j are calculated by :

$$\begin{aligned}\hat{w}_j &= (\alpha_j^w n_j/\Xi + (1-\alpha_j^w)w_j)\gamma \\ \hat{\boldsymbol{\mu}}_j &= \alpha_j^\mu E_j(x) + (1-\alpha_j^\mu)\boldsymbol{\mu}_j \\ \hat{\boldsymbol{\sigma}}_j^2 &= \alpha_j^\sigma E_j(x^2) + (1-\alpha_j^\sigma)(\boldsymbol{\mu}_j^2 + \boldsymbol{\sigma}_j^2) - \hat{\boldsymbol{\mu}}_j{}^2\end{aligned} \tag{7}$$

Here, γ is a scale factor that ensures that all the weights sum to one. $\alpha_j^w, \alpha_j^\mu, \alpha_j^\sigma$ are adaptation coefficients calculated from the data: $\alpha_j = \frac{n_j}{n_j+r}$, where r is known as *relevance factor*. If the relevance factor is extremly small, system would produce new speaker for the most of new utterances. It happens because new model become too adapted to new data and it is not general enough to recognize other data from the same speaker. If $r = 0$, then new model is trained only on the data from current utterance. In the opposite case, when the relevance factor is large, impact of adaptation data decreases. If we set $r = \infty$ then adapted model would be just the copy of UBM. This shows that relevance factor should be carefully chosen.

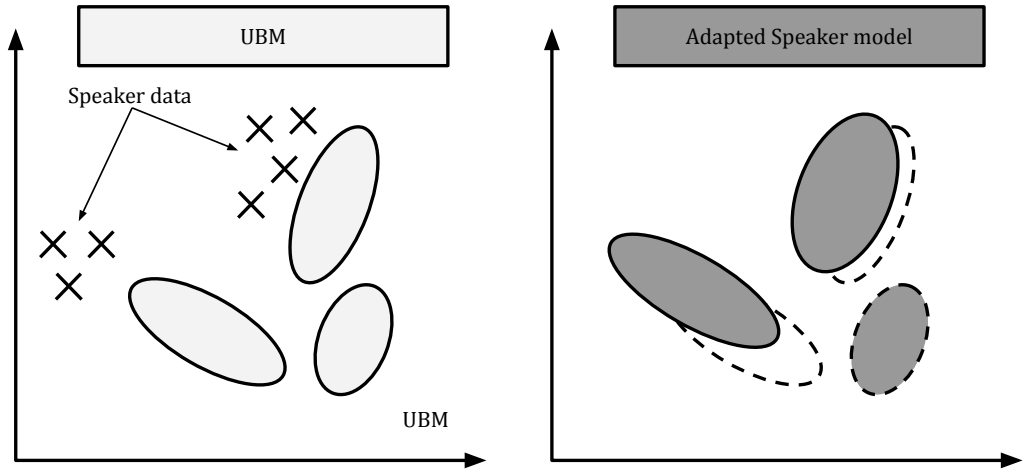

Figure 4: Graphical illustration of MAP adaptation process. Adopted from [25].

Although in general case means, covariances and weights should be adapted, in practice good performance can be achieved by adapting the mean vectors only [25]. Test segment should then be examined against UBM and the target speaker models. In further studies [10], the following idea was developed: adapt a pretrained UBM to new speech utterance and stack the mean vectors of resulting GMM. The result would be one *supervector* of length dK that represents the whole utterance and can be used as a new feature vector. Fig. 5 illustrates the process of creating GMM supervector in case of 2-dimensional feature vectors and UBM having 3 components.

The use of GMM supervectors has been found effective not only in speaker verification but

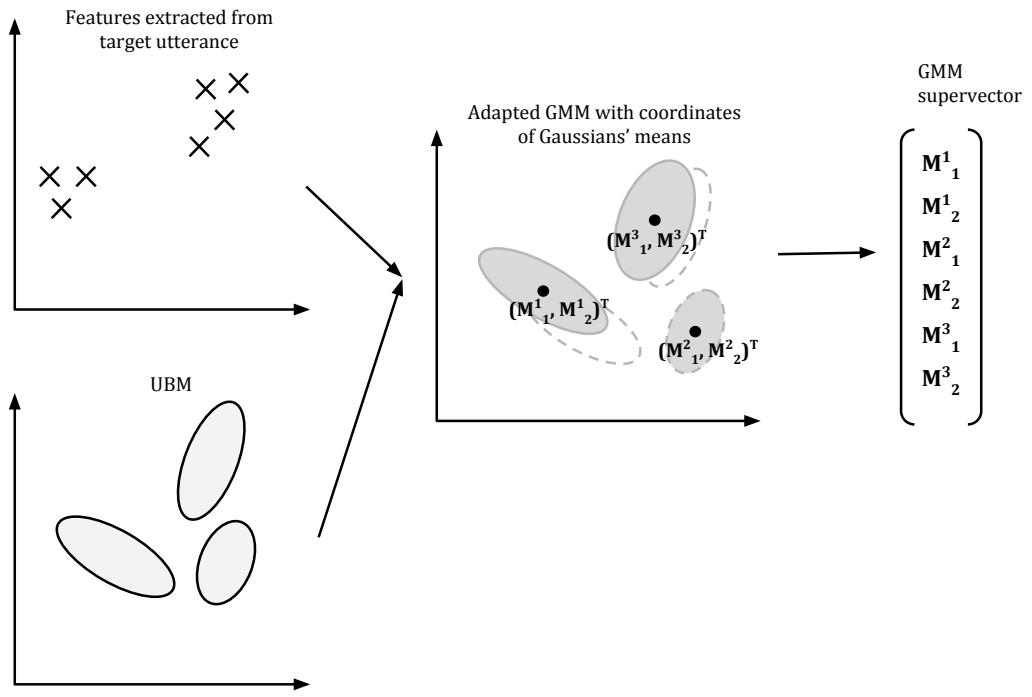

Figure 5: Illustration of retaining a GMM supervector. M^i denote coordinates of mean vector of i-th Gaussian.

also in other tasks, including age and gender recognition and fall detection [27, 28]. It has one obvious advantage over acoustic features such as MFCCs: for any utterance the length of this vector is fixed, although it is generally a large number. For instance, if the number of Gaussians, K, is 1024, and we use 12 base MFCC coefficients with energy, deltas and delta-deltas, the dimensionality of the GMM supervector will be $1024 \times 39 = 39936$.

3.3 Joint factor analysis

One of the main challenges in the speaker recognition is the presence of so-called *channel effect*. It means that every utterance contains not only information about the speaker but it is also affected by the characteristics of the microphone and environment. When building speaker models adapted from UBM, two utterances from the same speaker can be placed far away from each other in feature space because of differences in the recording setups. It would be good if some compensation would be applied to make sure that recordings of the same speaker are recognized as the same speaker even if they were made with different handsets.

Joint factor analysis (JFA) is one of the techniques aimed to deal with channel effect and perform channel compensation [29]. In this approach, a GMM supervector is represented as the

sum of two components: speaker s and channel c supervectors, as follows:

$$\boldsymbol{M} = \boldsymbol{s} + \boldsymbol{c}, \quad \text{where } \boldsymbol{c} = \boldsymbol{U}\boldsymbol{\phi} \text{ and } \boldsymbol{s} = \boldsymbol{m} + \boldsymbol{V}\boldsymbol{\psi} + \boldsymbol{D}\boldsymbol{\omega} \tag{8}$$

Here, $\boldsymbol{U}$ is known as the *eigenchannel matrix* and it defines session subspace, while $\boldsymbol{\phi}$ is a vector of *channel factors* estimated for a given speech segment. $\boldsymbol{m}$ refers to the UBM supervector and $\boldsymbol{V}$ and $\boldsymbol{D}$ jointly define the speaker subspace. $\boldsymbol{V}$ is known as the *eigenvoice matrix* and $\boldsymbol{D}$ is a diagonal *residual matrix*. Finally, ψ and $\boldsymbol{\omega}$ are speaker-dependent factors. The matrices $\boldsymbol{U}, \boldsymbol{V}$and $\boldsymbol{D}$ are the *hyperparameters* of the JFA model and they are estimated in advance by using labeled training utterances. Then, for each given utterance, parameters $\boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\omega}$ can be estimated. Finally, we can remove the session component and perform the scoring of test utterance against channel-compensated speaker model $\boldsymbol{M} - \boldsymbol{U}\boldsymbol{\phi}$. Detailed description how it can be done is given in [29]. Also, good source of practical implementation details of JFA is matlab tutorial made in Brno University of Technology [30].

3.4 Front-end factor analysis (i-vectors)

JFA described above is based on two different subspaces, channel and speaker, defined by matrices $\boldsymbol{U}$ and $\boldsymbol{V}$, respectively. But it was found in [4] that channel component contains some speaker information even if it was aimed at modeling only the session factors. This provided motivation for introducting a revised technique, *front-end factor analysis* [4], where factor analysis is viewed as a "feature extractor". In this method, there is no distinction between speaker and channel subspaces. With this assumption, GMM supervector can be rewritten as

$$\boldsymbol{M} = \boldsymbol{m} + \boldsymbol{T}\boldsymbol{v} \tag{9}$$

Here, $\boldsymbol{m}$ is a speaker and session independent supervector (in practice, the UBM supervector) and $\boldsymbol{T}$ is a low-rank matrix containing the eigenvectors with largest eigenvalues of total variability covariance matrix. The vector $\boldsymbol{v}$ defines total factors; it is a random vector with standard normal prior distribution. The posterior mean of $\boldsymbol{v}$ is known as *identity vector* or simply *i-vector*. This approach assumes that $\boldsymbol{M}$ has normal distribution with mean $\boldsymbol{m}$ and covariance $\boldsymbol{T}\boldsymbol{T}'$. Training of the matrix $\boldsymbol{T}$ is done in advance, but unlike training the hyperparameters of JFA, the training data does not have to be labeled.

In general, the i-vector model projects speech utterance onto a low-rank subspace so that each utterance can be presented by single a vector $\boldsymbol{v}$, which can be seen as a new feature vector. This representation has an obvious advantage over the supervectors: the dimensionality of i-vectors is much lower. Usually i-vectors are chosen to be 400 to 600-dimensional, which means that their processing requires less computation and memory.

Usage of i-vectors lead to considerable increase of recognition accuracy compared to other methods not only for the tasks of speaker recognition, but also tasks such as language [5] or accent [6] recognition. Thus, application of i-vectors to the problem of age estimation can produce good results as well.

3.5 Session compensation

An important consideration in speaker recognition is compensation effects of inter-session variability [31, 32]. Development of session compensation techniques is inspired by necessity to remove session variability from features. Doing so helps in focusing to the essential between-class information. Here, we discuss two widely applied techniques: *within-class covariance normalization* (WCCN) and *linear discriminant analysis* (LDA).

WCCN [31] is a commonly used technique for session compensation. It normalizes the average within-class covariance of feature space (e.g. i-vector space) to identity matrix. It helps to suppress the undesired directions of large within-class variation that dominate the feature space. WCCN transformation matrix $\boldsymbol{B}$ can be found by solving

$$\boldsymbol{B}\boldsymbol{B}' = \left[\frac{1}{J}\sum_{j=1}^{J}\frac{1}{N_j}\sum_{i=1}^{N_j}(\boldsymbol{v}_j^i - \bar{\boldsymbol{v}}_j)(\boldsymbol{v}_j^i - \bar{\boldsymbol{v}}_j)'\right]^{-1} \tag{10}$$

In (10), $\boldsymbol{v}_j^i$ is the i-th feature vector (e.g i-vector) of the j-th speaker, J is the total number of speakers and N_j is the number of vectors for j-th speaker. Finally, $\bar{\boldsymbol{v}}_j$ denotes mean feature vector of the j-th speaker. In practice, matrix $\boldsymbol{B}$ is derived through Cholesky decomposition. After training $\boldsymbol{B}$, all the training and future test vectors $\boldsymbol{x}$ are normalized by $\boldsymbol{x} \mapsto \boldsymbol{B}^T\boldsymbol{x}$.

The second technique, LDA [32], is used for session compensation and dimensionality reduction. It attempts to emphasize class discriminatory information by transformation of the feature space. For this reason, two variables are introduced: *between-class scatter* $\boldsymbol{S}_b$ and *within-class scatter* $\boldsymbol{S}_w$ (11). Here, $\bar{\boldsymbol{v}}_j$ still denotes mean feature vector for the j-th speaker and $\bar{\boldsymbol{v}}$ is the mean vector of all feature vectors in the training set.

$$\begin{aligned}\boldsymbol{S}_b &= \sum_{j=1}^{J}(\bar{\boldsymbol{v}}_j - \bar{\boldsymbol{v}})(\bar{\boldsymbol{v}}_j - \bar{\boldsymbol{v}})' \\ \boldsymbol{S}_w &= \sum_{j=1}^{J}\sum_{i=1}^{N_j}(\boldsymbol{v}_j^i - \bar{\boldsymbol{v}}_j)(\boldsymbol{v}_j^i - \bar{\boldsymbol{v}}_j)'\end{aligned} \tag{11}$$

To train the LDA transformation matrix, eigenvalue decomposition for scatters $\boldsymbol{S}_w^{-1}\boldsymbol{S}_b\boldsymbol{e} = \lambda\boldsymbol{e}$ is first carried out. The projection matrix is then composed of the k eigenvectors corresponding to the k largest eigenvalues λ. k is a predetermined parameter, the target dimensionality.

3.6 Support vector machine and support vector regression

Here, we briefly introduce the *support vector regression* (SVR) technique. SVR is a regression extension of the widely known *support vector machine* (SVM) classifier. Let us first briefly recall the basic principles of SVM.

The core idea of SVM [33] is to find a hyperplane in the feature space that separates two classes with maximum margin. To formalize this idea, we introduce the following notations: we are given a training set $S^{tr} = \{(\boldsymbol{a}_1, b_1), ..., (\boldsymbol{a}_P, b_P)\}, \boldsymbol{a}_p \in \mathbb{R}^d, b_p \in \mathbb{R}$. $\boldsymbol{a}_p$ is the input of the model, b_p is the desired output. In the classification task, b_p is either +1 or -1. The aim is to build a function $f(\boldsymbol{a})$ so that it will accurately predict output. In SVM, we seek for a linear sepator of the form $f(\boldsymbol{a}) = \boldsymbol{\eta}'\Phi(\boldsymbol{a}) + z$, where $\Phi(\boldsymbol{a})$ denotes some mapping function to a higher-dimensional (Hilbert) space, $\boldsymbol{\eta}$ is a weight vector and z is a bias term. To train the model, one finds $\boldsymbol{\eta}$ and z that maximize the following objective function [11]

$$\frac{1}{\|\boldsymbol{\eta}\|} \min[b_p(\boldsymbol{\eta}'\Phi(\boldsymbol{a}_p) + z)] \tag{12}$$

This objective function presents perpendicular distance from the separating hyperplane to the training example $\boldsymbol{a}_p$. This problem can be transfered to the task of minimization of $\frac{1}{2}\|\boldsymbol{\eta}\|^2$ with inequality constraints $b_p(\boldsymbol{\eta}'\Phi(\boldsymbol{a}_p) + z) \geq 1, p = 1, .., P$. This problem is solved by introducing Lagrange multipliers α_n. The details how it is done in practice one can find e.g. in [33]. Then, the optimal solution can be shown to be of form $f(\boldsymbol{a}) = \sum_{n=1}^{N} \alpha_n b_n K(\boldsymbol{a}, \boldsymbol{a}_n) + z$. Here, $K(\boldsymbol{a}, \boldsymbol{a}_n) = \Phi(\boldsymbol{a})\Phi'(\boldsymbol{a}_n)$ is known as the *kernel function*. A 2-dimensional illustration of SVM classification is shown on Fig. 6.

The idea of SVMs was later generalized for regression tasks, where the goal is to build a hyperplane for which maximum of training data points would lie no further than ϵ distance from this hyperplane [34] (Fig. 7). Here, we modify the previous notation slightly. Now b_p can be an arbitrary real number not restricted to be discreate like in the case of classification. We again seek for a linear function of the form $f(\boldsymbol{a}) = \boldsymbol{\eta}'\Phi(\boldsymbol{a}) + z$ and solve optimization problem (13) with inequality constraints (14).

$$\frac{1}{2}\|\boldsymbol{\eta}\|^2 + C\sum_{n=1}^{N}(\xi_n + \hat{\xi}_n) \tag{13}$$

$$\begin{cases} b_n - \boldsymbol{\eta}'\Phi(\boldsymbol{a}_n) - z \leq \epsilon + \xi_n \\ \boldsymbol{\eta}'\Phi(\boldsymbol{a}_n) + z - b_n \leq \epsilon + \hat{\xi}_n \\ \xi_n, \hat{\xi}_n \geq 0 \end{cases} \tag{14}$$

Here, ξ_n and $\hat{\xi}_n$ are so-called *slack variables* which will vanish in the optimization pro-

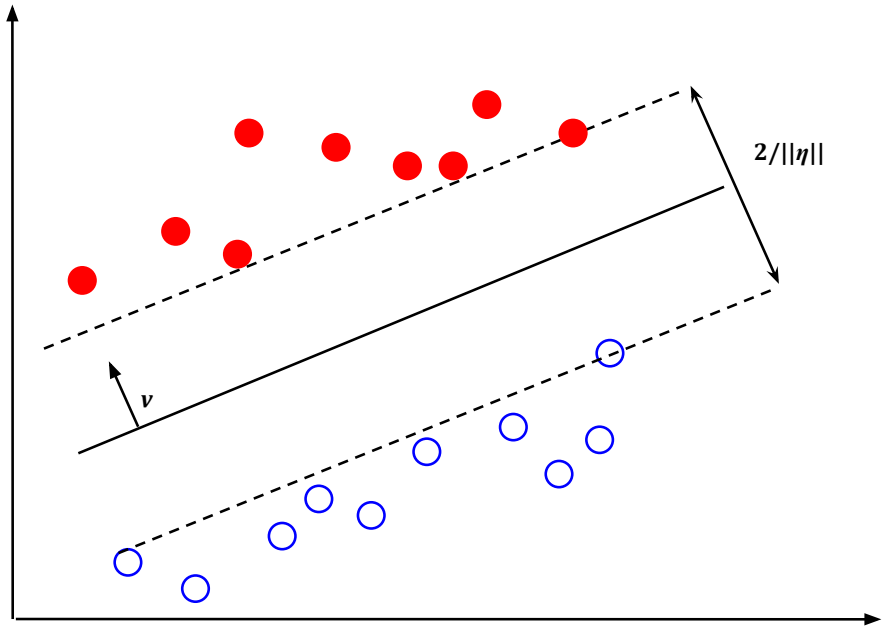

Figure 6: Support vector machine classification in 2-dimensional case.

cess. Slack variables are needed to cope with possible infeasible constraints of the optimization problem. C is positive *regularization parameter* determining the trade-off between cost of deviations of function $f(\boldsymbol{a})$ larger than ϵ for training data and its flatness. The problem can be efficiently solved by introducing Lagrange multipliers $\alpha_n, \hat{\alpha}_n, \mu_n, \hat{\mu}_n$ and optimizing the following Lagrangian L.

$$L = C\sum_{n=1}^{N}(\xi_n + \hat{\xi}_n) + \frac{1}{2}\|\boldsymbol{\eta}\|^2 - \sum_{n=1}^{N}(\mu_n\xi_n + \hat{\mu}_n\hat{\xi}_n)$$
$$- \sum_{n=1}^{N}\alpha_n(\epsilon + \xi_n - b_n + \boldsymbol{\eta}'\Phi(\boldsymbol{a}_n) + z) - \sum_{n=1}^{N}\hat{\alpha}_n(\epsilon + \hat{\xi}_n - \boldsymbol{\eta}'\Phi(\boldsymbol{a}_n) - z + b_n)$$

It can be shown [34] that the solution to this optimization problem has the form $f(\boldsymbol{a}) = \sum_{n=1}^{N}(\alpha_n - \hat{\alpha}_n)K(\boldsymbol{a}, \boldsymbol{a}_n) + z$, where the parameters α_n and $\hat{\alpha}_n$ can be found through solving a *dual* optimization problem. $K(\boldsymbol{a}, \boldsymbol{a}_n)$ has the same meaning as in the case of SVM. More datails on computing the parameters α and z can be found in the original paper that introduced SVR [34]. In general, both the primal and dual problems are convex and the solution can be efficiently found by numerical methods.

In practice, SVM and SVR are realized in various software packages, such as LibSVM [35] and SVMTorch [36]. In the experimental part of this thesis, we use LS-SVMlab package

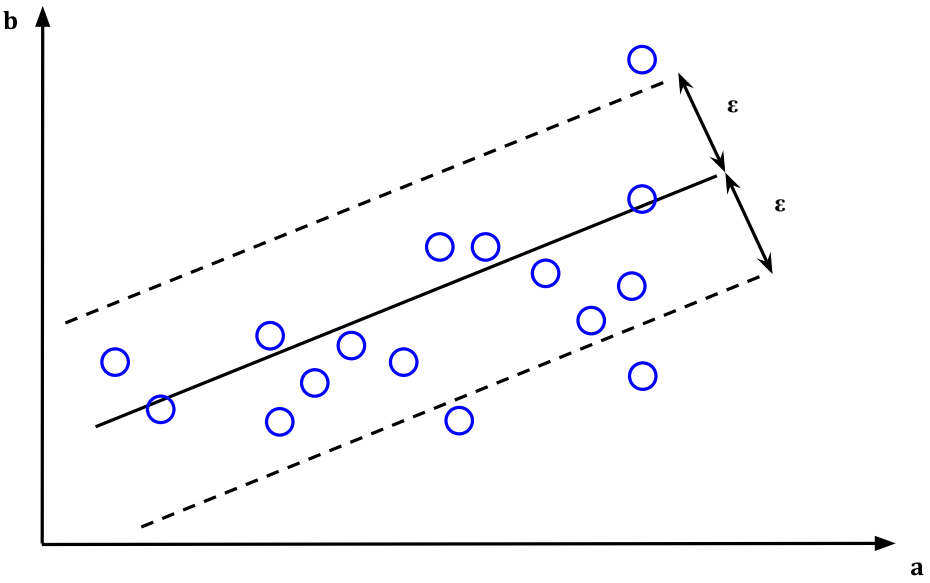

Figure 7: Support vector regression.

[37], which implements modification of SVM methods known as *least squares support vector machine* [38].

3.7 Neural networks

This section provides a short review of a very powerful approach used for function approximation, classification and other tasks, *artificial neural network* (ANN) [11]. In particular, we examine the most popular type of networks, *multilayer perceptron* (MLP).

Neural network is often presented as a system of interconnected units, known as *neurons*. Every such unit performs transformation of its input using some function and passes this value with some weight to the next neuron. Formally, one neuron is described by the following equation: $b = f(\boldsymbol{\eta}'\boldsymbol{a} + z)$. Here, $\boldsymbol{a}$ is the input vector, $\boldsymbol{\eta}$ are the weights, z is a bias term (we can drop it by adding to input vector one more value set to be 1 and to weight vector value z). Function $f()$ is the *activation function*. It is not restricted to have any specific form, but in this thesis we use either *hyperbolic tangent* $f(a) = \tanh(a)$ or *logistic function* $f(a) = 1/(1+e^{-a})$.

We can combine individual neurons in a structure shown in Fig. 8. The neurons are organized in larger units, *layers*. Neurons within each layer are not connected to each other in MLP but they are connected to other neurons from other layers through their layer inputs and outputs. The first and the last layers of network are called *input* and *output* layers, respectively, while

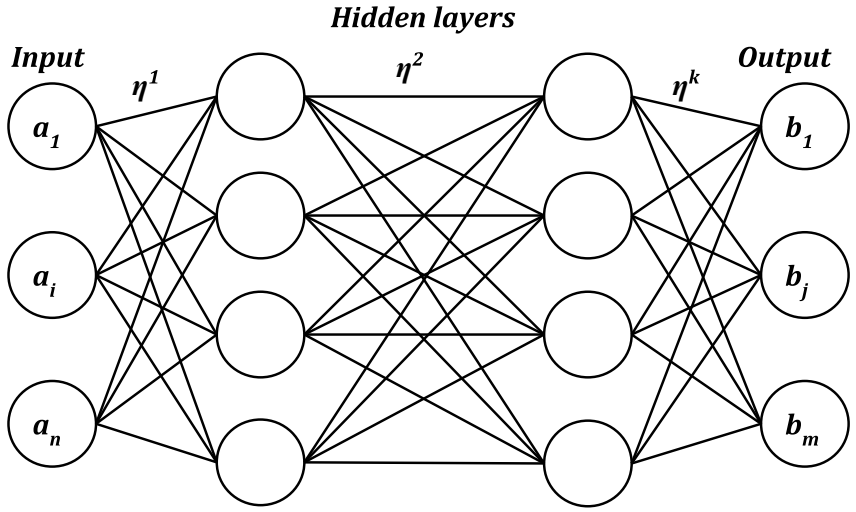

Figure 8: Multilayer perceptron. Adopted from [11].

all the others are *hidden* layers. The network operates as follows. When it has some input, it calculates its weighted sums and passes these sums to the corresponding neurons (every neuron has its own set of weights) of the first layer which performs their transformation and passes the results with new weights to the next layer and so on. The last layer provides the total output of the network for a given input.

Neural networks can be trained to model a given set of data by modifying its weights. Nowadays, many algorithms are available for training neural networks. The training objective is to minimize a suitable error function by modifying the network weights. Many techniques utilize gradient information of the error surface. The so-called *backpropagation* algorithm [39] is an efficient method for calculating the error gradient in the given point. It allows evaluating the gradients for each layer of the network iteratively.

In this thesis we consider two training methods, *stochastic gradient descent* (SGD) [40] and *Broyden–Fletcher–Goldfarb–Shanno* algorithm (BFGS) [41]. The first one is based on the fact that the total error for the whole training set, E, is the sum of errors for individual independent training cases, $E = \sum_{p=1}^{P} E_p$. While the standard gradient descent algorithm would update weights at iteration τ based on gradient for the whole data (15), SGD updates them according to gradients of one data point at a time (16). The parameter α in both equations denotes step size to the direction of the negative gradient. In this thesis, we divide data into several batches and weight update is done according to gradient of the error in every batch. We still call this algorithm SGD as opposed to basic gradient descent algorithm.

$$\boldsymbol{\eta}^{(\tau+1)} = \boldsymbol{\eta}^{\tau} - \alpha \nabla E(\boldsymbol{\eta}^{\tau}) \tag{15}$$

$$\boldsymbol{\eta}^{(\tau+1)} = \boldsymbol{\eta}^{\tau} - \alpha \nabla E_n(\boldsymbol{\eta}^{\tau}) \tag{16}$$

The SGD training algorithm is based only on the information about first derivative of the error function while BFGS requires second order derivative information as well. It belongs to the class of so-called *quasi-Newton* methods [42]. BFGS does *not* explicitly calculate the actual Hessian matrix of the second derivatives but makes an approximation, $\boldsymbol{H}$, and uses this during optimization. Weights are updates as follows. We search for the direction $\boldsymbol{p}^{\tau}$ as a solution to equation (17), and make a step α in this direction to the next weight point $\boldsymbol{\eta}^{(\tau+1)} = \boldsymbol{\eta}^{\tau} + \alpha \boldsymbol{p}^{\tau}$.

$$\boldsymbol{H}^{\tau} \boldsymbol{p}^{\tau} = \nabla E_p(\boldsymbol{\eta}^{\tau}) \tag{17}$$

In the experimental part of this thesis we use modification of BFGS algorithm called *limited memory BFGS*. Details of it can be found, for example, in [43].

One of the common problems in any machine learning task is to avoid *overfitting*. It means that we want to prevent neural net to memorize training set and perform perfectly on it but to have very poor generalization ability. There are various methods designed for neural networks to cope with this trade-off, such as *early stopping* [44] and *dropout* [45]. In this thesis, we adopt ℓ_2- regularization [46] which adds to the objective function an additional quadratic term that penalizes for large weights. Then, the objective function to be minimized takes a form $E + \frac{\lambda}{2} \sum_i \eta_i^2$, where λ is regularization parameter indicating relative importance of error depending on the data and regularizer. This technique helps to keep weights small unless they have large error derivatives. It leads to smoother model and helps in improving generalization since it stops network from fitting sampling error of training data.

4 Baseline method and experimental set-up for age estimation

In this Section we describe our baseline system for age estimation. In two papers [7, 8], Bahari and his co-authors introduced a method using i-vectors, method widely used in speaker recognition field, for the task of speaker age estimation. In some previous studies SVR was successfully used for age recognition [9]. Because of that successful experience with SVR, the authors decided to use it again, but apply it in combination with i-vectors. We describe this approach in more detail below.

4.1 Support vector regression

In the baseline approach, we use a modification of the standard SVM [33] method known as *least square* SVM (LSSVM) [38]. It has a faster training process than the standard SVM because it requires solving a system of linear equations instead of a quadratic programming problem. Further, it has fewer parameters to tune. It can also be extended to regression tasks. In the baseline paper one experiment comparing SVR and LSSVR was conducted, where the modified approach was found more accurate in terms of *mean absolute error* for NIST 2008 and NIST 2010 data sets (the advantage was not significant, so it is not clear if it will still be relevant for other experiment setups). Because of higher prediction accuracy and advantages in training process, Bahari and his colleagues decided to use LSSVR in their system. In this thesis, we take this approach as a baseline.

4.2 I-vector framework

For the i-vector extraction standard procedure described in Section 3.4 was used. After the training of an i-vector extractor, every speech utterance can be represented by its i-vector. Bahari and co-authors [7] found that it can be beneficial to use i-vector session-compensation. It aims at removing session variability from the i-vectors in order to better focus on between-class information. In both papers, Bahari *et al.* applied *within-class covariance normalization* (WCCN) [31]. Although experiments with WCCN are presented in both papers [7, 8] describing their baseline approach, it was applied differently. In the first work [7], WCCN was used to normalize covariance within different age classes. This approach was found unsuccessful, and in the second paper it was reconfigured so as to normalize *within-speaker* covariance instead. This strategy appeared to be more effective and helped to improve the accuracy of age estimation.

4.3 Procedure of training and testing

The principle proposed in [8] is explained on the scheme shown in Fig.9. In the training phase, i-vectors for each utterance are extracted, then WCCN is applied to suppress session variability effects. Further, the normalized i-vectors are presented to SVR together with the corresponding age labels to train it. In the run-time stage, an i-vector is extracted, WCCN transformation is applied and the normalized vector is fed to the regression model, which produces a predicted age (a scalar).

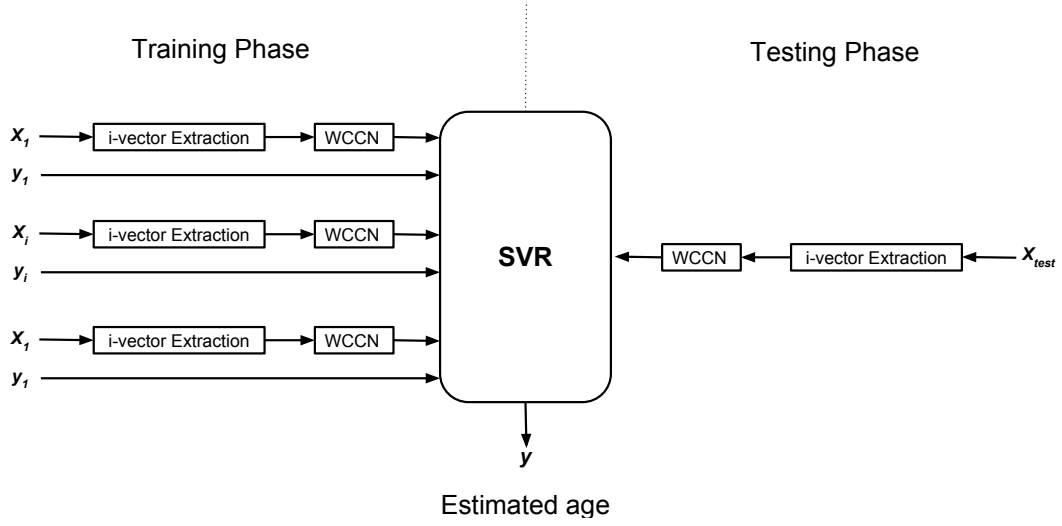

Figure 9: Schema of baseline approach. Adopted from [8]. x_i, y_i denote speech utterance and respective age label.

4.4 Performance measure

The original papers [7, 8] use two objective measures of age estimation accuracy. The first one is *mean absolute error* (MAE), calculated using the following formula :

$$\mathrm{MAE} = \frac{1}{N^{\mathrm{test}}} \sum_{n=1}^{N^{\mathrm{test}}} |\hat{Y}_n - Y_n|.$$

Here, N^{test} denotes the number of test segments, $\hat{Y}_n$ is the predicted age by the regression model and Y_n is the actual chronological age that serves as the ground truth. Smaller MAE is considered better. The second measure is *Pearson's correlation coefficient* between the vectors of estimated and chronological ages:

$$\rho = \frac{1}{N^{\mathrm{test}} - 1} \sum_{n=1}^{N^{\mathrm{test}}} \left(\frac{\hat{Y}_n - \mu_{\hat{Y}}}{\sigma_{\hat{Y}}} \right) \left(\frac{Y_n - \mu_Y}{\sigma_Y} \right).$$

Here, $\mu_{\hat{Y}}$ and $\sigma_{\hat{Y}}$ denote, respectively, the mean and standard deviation of estimated speaker's age. Similarly, μ_Y and σ_Y correspond to the same measures of actual age. Higher ρ is considered better. Throughout this thesis, we adopt the same performance metrics as in the baseline approach.

4.5 Data and experimental setup

The data used for experiments in the baseline approach is NIST speaker recognition evaluation (SRE) datasets from year 2008 and 2010. These corpuses were selected because they contain utterances from large amount of speakers and rich metadata, including speaker's age. Some restrictions were made for the used data: firsty, only telephone data was selected. Secondly, we considered utterances from speakers between 20 to 70 years old only. The reasoning for these choices are as follows. Telephone recordings were selected because in these corpora, recordings of other type contain talk from multiple speakers, which would cause difficulties in both model training and measuring performance. Discarding of younger and older speakers, in turn, is made because they are too few compared to speakers of other ages. Table 2 presents summary of the used data. Figure 10 further shows the age distribution of male and female speakers of the selected utterances from NIST 2008 and NIST 2010 datasets.

Table 2: Summary of the data used for age estimation

	NIST 2008		NIST 2010	
	Male	Female	Male	Female
Number of speakers	412	742	218	224
Number of utterances	1402	2457	2560	3023
Quality	Telephone			
Sampling rate	8.0 kHz			

For the experiments, all the data used for age estimation was divided into 15 folds so that there is no speaker overlap across the folds. Then, 15 independent tests were run, in each of them 14 folds were used as the training set and the 15th as an independent (held-out) test set. The final result (MAE and ρ described before) presented is the average of those metrics respectively for all the 15 tests. Each time, two independent age recognition systems are built for male and female speakers and the results also are presented for male and female separately.

This thesis uses the same experimental set-up. The only difference is that for training UBM and total variability subspace matrices, we use all the available NIST SRE corpora except the ones used for evaluation (NIST 2008 and NIST 2010). In the original papers [7, 8], the authors used only NIST 2004-2006 datasets. As we will see later, that did not affect final accuracy and the results we get, when replicating the baseline approach, are very close to the results reported

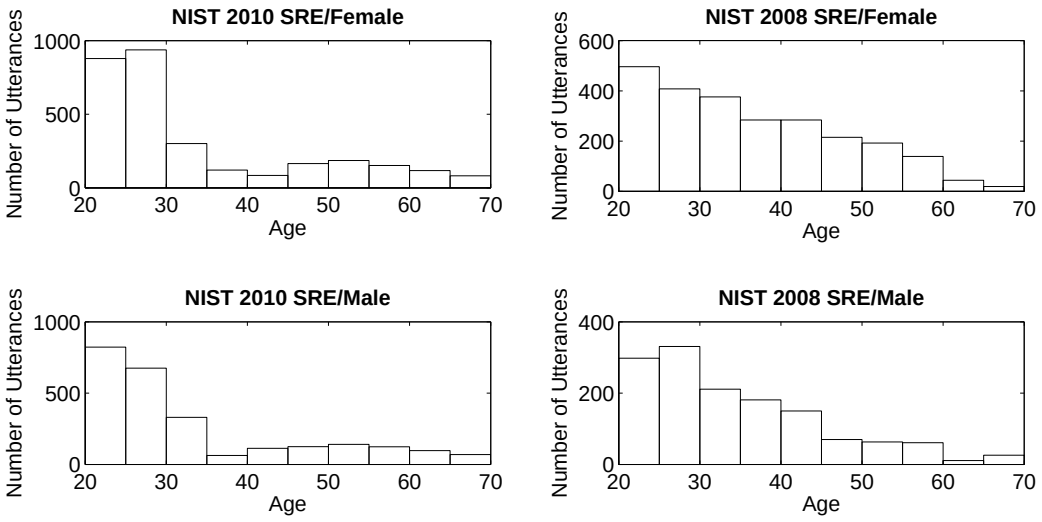

Figure 10: Age histogram of the selected speech utterances from NIST 2008 and 2010 corpuses

in the origal paper [8]. Thus, we have confidence that our baseline is correctly implemented and represents the correct state-of-the-art of age estimation.

5 Alternative features

I-vectors based on mel-frequency cepstral coefficient (MFCC) features were found effective for age estimation in [7, 8]. But what if some other acoustic features are better suited for age estimation? In this Section, we will try to find these new features. First of all, we will study how different set-ups for i-vector extraction affect the recognition results. In the following two subsections, we consider two other options alternative to i-vectors. In both cases, neural networks are utilized, but the usage of the networks are different. Section 5.2 describes experiments with MFCC features and usage of neural network as a regressor. Section 5.3, on the other hand, demonstrates the use of neural network as a feature extractor.

5.1 I-vector extraction setups

First idea which seems worth to explore is different initial settings for i-vector extractor. The first stage in obtaining i-vectors is extraction of acoustic features (MFCC features are one option but there are also several other techniques). The resulting i-vectors are of course strongly dependent on the type of acoustic features used to construct them. Five differents variants of features are studied in this thesis. Some of them are modifications of MFCCs while the others are completely different types.

We first describe the setup for i-vector extraction used in the baseline approach.We use 60-dimensional MFCC vectors (19 coefficients with energy, deltas and delta-deltas) as the acoustic features. Then short-term *cepstral mean and variance normalization* (CMVN) [47] is applied. CMVN is one of the most common techniques in speech and speaker recognition. CMVN is described by equation (18).

$$\begin{aligned} \hat{C}_t &= \frac{C_t - \mu_t}{\sigma_t}, \\ \mu_t &= \frac{1}{L} \sum_{n=t-\frac{L}{2}}^{t+\frac{L}{2}-1} C_n, \\ \sigma_t^2 &= \frac{1}{L} \sum_{n=t-\frac{L}{2}}^{t+\frac{L}{2}-1} (C_n - \mu)^2 \end{aligned} \tag{18}$$

Here, μ_t denotes mean value of features on the given utterance segment of length L and σ_t is standard deviation on the same segment. $\hat{C}_t$ and C_t are the original and the modified value of the feature, respectively.

Applying this transformation produces a new set of features with zero-mean and unity variance. The length of normalization segment is not strictly specified but it can be chosen according to the task and properties of the original signal. One must find a suitable balance, since too short segments do not have enough information specifying mean and variance. On the other hand,

too long segments can result in better estimates but cause longer delays during processing: we need to wait until the end of each segment to compute the needed statistics. Based on the normalized features, we extract 400 dimensional i-vectors. Table 3 shows the baseline results we obtained using described approach along with the results reported in [8]. The results are close to each other and differences likely caused by differences in random division into 15 folds and UBM/i-vector data selections. Further, when referring to baseline we addres the results of the system that we implemented.

Table 3: MAE (in years) and ρ for baseline approach

	Male		Female	
	MAE	ρ	MAE	ρ
Our baseline	6.65	**0.73**	**5.75**	**0.80**
Baseline in [8]	**6.53**	**0.73**	5.78	**0.80**

The first experiment looks for dependence of the i-vector dimensionality and recognition accuracy because it is likely that vectors with higher number of elements may contain more age-dependant information and be helpful in improving the age estimator's performance. Here, we compare baseline setup with 600-dimensional i-vectors based on exactly the same MFCC vectors and trained support vector regression estimator on those. 600-dimensional i-vectors were chosen as alternative to baseline since they are widely used in speaker and language recognition systems [48] and shown good level of performance in several tasks. The other possible

Table 4: MAE (in years) and ρ for SVR age estimators trained on i-vectors of various sizes

	Male		Female	
I-vector Setup	MAE	ρ	MAE	ρ
baseline (400-dim i-vectors)	6.65	**0.73**	5.75	**0.80**
600-dim i-vectors	**6.59**	0.72	**5.76**	**0.80**

option is to study how CMVN affects performance of the system. To this end, we compare the baseline approach with i-vectors built on unnormalized MFCCs. Also, we consider another possible transformation of the feature vectors, utterance mean normalization. It is special case of CMVN, when normalization segment lenght L equals utterance length and variance normalization is discarded. In this approach sentence utterance is subtracted from the features, which means that new features have zero mean for each utterance.

Shifted delta cepstral coefficients (SDCs) [49] represent another option for feature selection

and can be expressed in the following way:

$$s_{i\theta+j}(t) = C_j(t + i\delta + \chi) - C_j(t + i\delta - \chi), i = 0, .., \kappa - 1.$$

Here, $C_j, j = 1, .., \theta - 1$ are the base MFCC coefficients. We see that SDCs are defined by four parameters. The first one, θ, defines the number of the cepstral coefficients. The second parameter, χ, determines time difference between the frames, δ is time shift between two blocks and κ is the number of blocks. So, SDCs are basically κ blocks of delta cepstral coefficients. SDCs add contextual temporal information to the feature vectors which can be useful in terms of age estimation.

The last features are so-called *Perseus* features. Generally, Perseus features are based on "MMeDuSa" features [50]. Specifically, they are a combination of MFCCs and MMeDuSa features. Several modifications in MFCC extraction procedure described in section 3.1 are made to obtain the Perseus features. The first change is usage of *gammatone* filter bank [51] instead of mel filter bank. Another modification is adding to the feature vector sub-frame energy estimations. Finally, we replace logarithmic compression used in MFCC extraction by 1/15-th root compression.

In the two last experiments, we built i-vectors based on these two alternative types of acoustic features. The results of described experiments are presented through Tables 4 to 6.

Table 5: MAE (in years) and ρ for SVR age estimators trained on i-vectors with various normalization

	Male		Female	
I-vector Setup	MAE	ρ	MAE	ρ
baseline (short-term CMVN)	**6.65**	**0.73**	**5.75**	**0.80**
no feature normalization	6.77	0.70	5.85	0.79
utterance CMVN	6.73	0.72	5.78	**0.80**

Table 6: MAE (in years) and ρ for SVR age estimators trained on i-vectors built on alternative acoustic features

	Male		Female	
I-vector Setup	MAE	ρ	MAE	ρ
baseline (MFCC features)	**6.65**	**0.73**	**5.75**	**0.80**
Perseus features	7.01	0.69	5.99	0.79
SDC features	6.72	0.72	6.01	0.78

The best performance, in terms of age estimation, is achived with baseline schema of i-vector extraction. I-vectors with increased dimensionality yield almost the same results, but

require more computation. All the other experimented ideas degrade accuracy.

5.2 MFCC as direct features of age estimator

Since different setups for i-vector extraction did not show any improvement in age estimation, performance of other types of features is worth exploring. The first features that we examine are MFCCs. Being low-level acoustic features, MFCCs potentially contain useful information for age estimation. Here we conduct experiments with neural networks instead of SVR. The common schema of these experiments is as follows. At the training phase for every utterance, all MFCC vectors are considered independent. The network gets a single vector as an input, and the label for it is the same as the label for the whole utterance. Neural network is trained on those vectors. During the test phase we predict age label for every single feature vector from one utterance and average those results across the utterance to get the final age label for it. This schema is depictured on the Fig. 11.

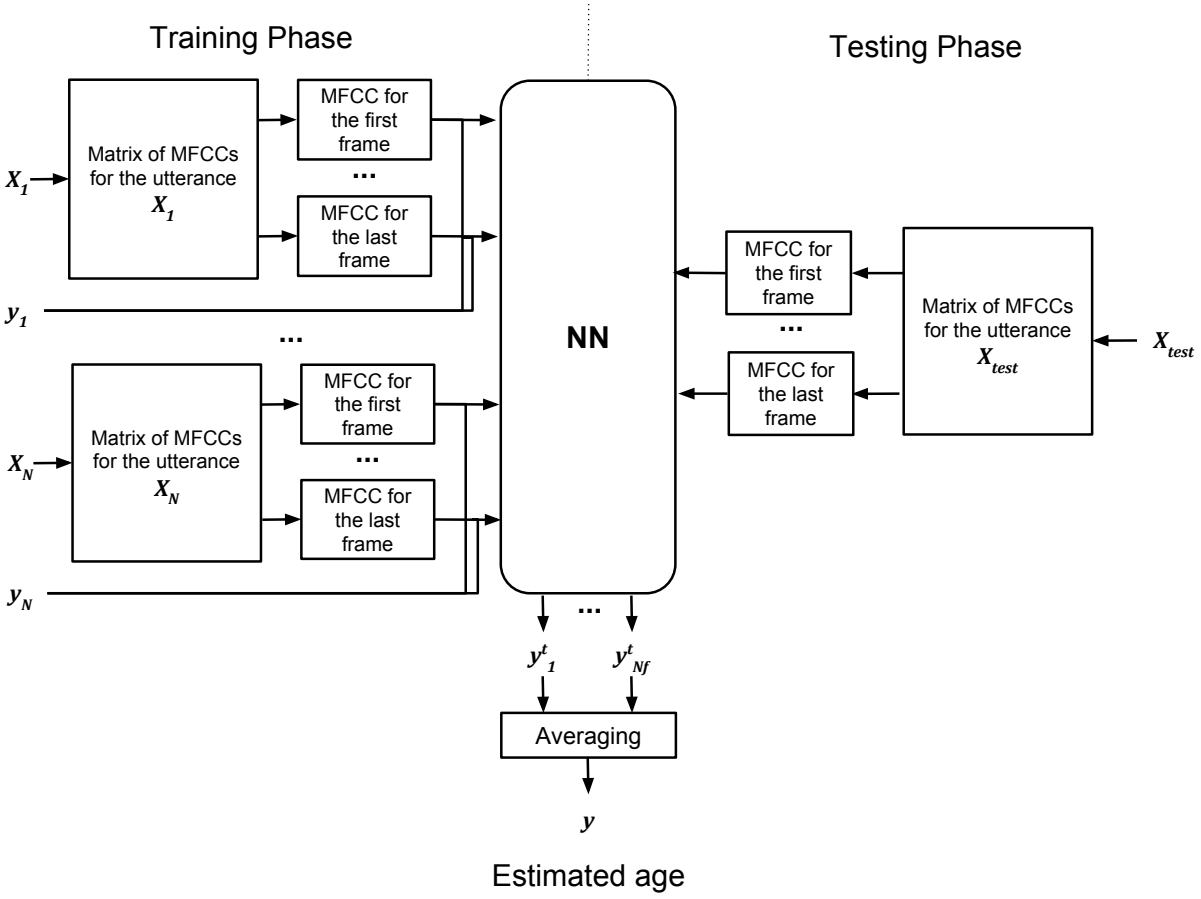

Figure 11: Schema of using MFCC features in combination with neural network for age estimation. x_i, y_i denote speech utterance and respective age label, y_i^t denote predicted age for frames of test utterance, y is the final age label for the whole test utterance.

The described procedure was repeated in two variants: the first one was exactly the same as written above, the second one has a major modification. Instead of taking MFCC vectors separately we stacked consecutive 9 MFCC frames together to one vector of higher dimensionality. For each utterance, those vectors were further used for age prediction. Motivation for taking stacked MFCCs is to add temporal information to the feature vector, similar to SDCs.

Due to large amount of data, training of the network requires time and computational resources. For this reason, we conducted preliminary experiments only for female speakers. Table 7 shows the performance of neural networks trained on single and stacked MFCCs for females. In both cases, network has a single hidden layer with 512 neurons and is trained with SGD method [40] presented in Section 3.7.

Table 7: MAE (in years) and ρ for the neural network age predictor trained on MFCC features for female speakers

	Female	
Features	MAE	ρ
single MFCCs	10.85	0.47
9 stacked MFCCs	**9.79**	**0.53**

Comparing Tables 4 and 7, the results on direct features are way behind the performance of i-vector system. This might be because of high variations of single acoustic feature vectors corresponding to the same utterance and the same age label. Stacking MFCC vectors together helped to slightly improve performance but it is still far from the i-vector system results. We do not develop this direction of experiments further.

5.3 Statistical feature extraction using neural networks

As another alternative to i-vectors, we examine new types of features. The core idea is that a trained neural network nonlinearly compresses the input information. This way, the trained neural network can be utilized for nonlinear feature extraction.

Features that we use are similar in spirit to the so-called *bottleneck* features [52]. Bottleneck features are computed as follows. They are outputs of some hidden low dimensional layer of network trained to reproduce the input vectors, called an *autoencoder* [52]. That is, the lower dimensional layer compresses data but all the essential information is retained. Usage of that compressed version can be fruitful since the input dimensionality is decreased while main variations are retained.

In our case, instead of training a self-predicting network, we train a neural network to predict age as in the section 5.2. MFCCs are used as input vectors to this network. But, instead

of making prediction for the test data using this network, we apply it as a feature extractor. The procedure to train and test the age predictor is formalized in the following procedure.

- **Training**
  1. Train a neural network as described in Section 5.2 to predict age from MFCC vectors and store the network
  2. For each training utterance:
     (a) Present all the training MFCCs for the given utterance one by one and store the outputs of the network's hidden layer
     (b) Average outputs of hidden layer across the whole utterance
     (c) Store the result of previous step as new feature vector
  3. Train any regression model (e.g. neural network or SVR) on the obtained features to predict age
- **Testing**
  1. Present all MFCCs of the test utterance one by one to the trained network and store the outputs of network's hidden layer
  2. Average the outputs of hidden layer across the whole utterance
  3. Store the result of the previous step as a new feature vector
  4. Using the regression model back-end and the new feature vector to predict the age label for the test utterance

Table 8: MAE (in years) and ρ for the SVR age predictor trained on neural network's statistical features for female speakers

	Female	
Kernel function of SVR	MAE	ρ
Linear	**7.06**	**0.70**
RBF	7.53	0.66

In the following experiments, we trained a neural network with 1024 neurons in the hidden layer with sigmoid activation functions. Then new data was fed to the trained network and 1024-dimensional feature vectors were obtained using the process described above. As a regression

model we examine SVR. As two alternatives we took two different kernel functions for SVR: *linear* and *radial basis function*. The results achieved with the two types of SVRs are presented in Table 8. Here we will see only the results for female speakers similar to the previous section.

This new type of features greatly outperforms MFCC based age estimators in Table 7. It is still, however, behind the baseline i-vector approach (Table 4).

6 Neural networks for i-vector based regression

Since training different features (Section 5) did not appear a promising direction, our next step is to fix the features to i-vectors and study the role of the back-end regression method. Most of the previous works in age estimation were using SVM in case of classification or SVR for regression. This section studies the use of neural networks for age estimation. Here we used only MFCC-based 400-dimensional i-vectors which were found the best i-vector setup when using SVR (details in Section 5.1). For the majority of our experiments, we use MLP with a single hidden layer. Penalties (details in Section 3.7) of 0.1 and 0.01 for the weights of the first and second layers are used, respectively. The learning rate in all experiments is set to 0.5. These values were optimized in initial experiments utilizing on MLP with 512 neurons in the hidden layer.

6.1 Effect of the training algorithm

In the first experiment, we train a single hidden layer neural network and study the effect of the optimisation method. Fig. 12 shows dependency of MAE and ρ on the number of neurons in the hidden layer for two optimisation methods: *stochastic gradient descent* (SGD) [40] and *Broyden–Fletcher–Goldfarb–Shanno* (BFGS) [41] algorithm. These results are shown for male speakers only, for females the trends are similar.

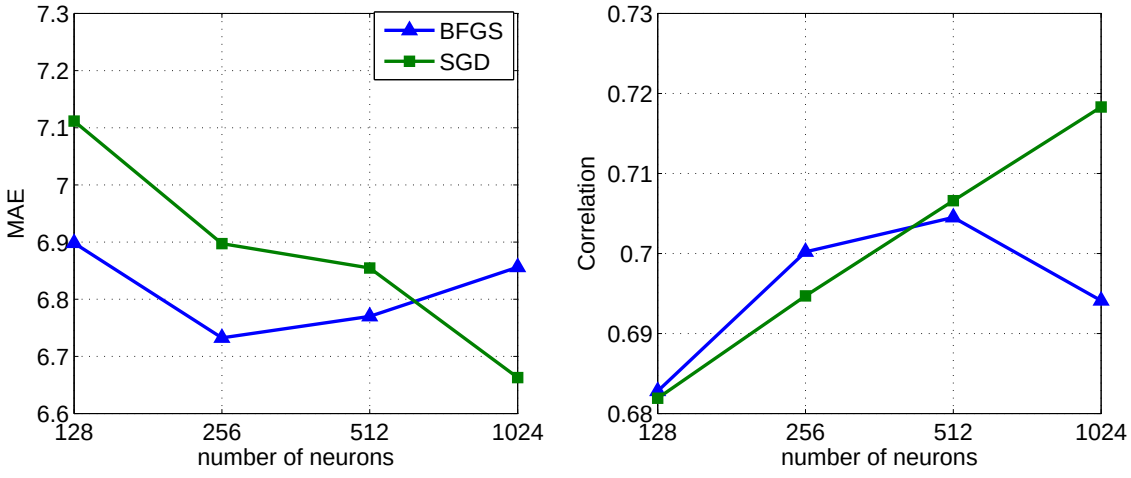

Figure 12: Mean average error for age estimation of male speakers.

As Fig. 12 shows, when using BFGS for training, the size of the hidden layer does not affect performance as much as when SGD is used. When the number of neurons is low, BFGS outperforms SGD, but for larger network, the second method yields higher accuracy and also it is faster than BFGS.

6.2 Effect of WCCN

As mentioned in Section 4.2, Bahari *et al.* found WCCN applied to i-vectors to improve SVR-based age estimator performance. Interestingly, [8] reported that improvement was achieved when each speaker was treated as a different class. The supposedly more meaningful strategy, training WCCN using discrete age classes, actually decreased performance of age estimator in [7]. For this reason, and since our baseline results (Table 3) are similar, we study WCCN using speaker labels only for convenience. When implementing the neural network approach, we were curious to study usefulness of WCCN, too. Tables 9 and 10 compare the impact of WCCN for networks of various sizes. The same single hidden layer architecture for networks is used as above (Section 6.1) and the training algorithm is SGD. For large networks (512 and 1024 hidden units), WCCN helps and will be used in all the remaining experiments.

Table 9: MAE (in years) and ρ of the neural network age estimator with and without WCCN for female speakers.

	no WCCN		WCCN	
Size of hidden layer	MAE	ρ	MAE	ρ
128	**6.19**	**0.77**	6.32	0.76
256	**5.93**	**0.78**	6.12	0.77
512	5.91	0.78	**5.72**	**0.80**
1024	5.66	0.80	**5.49**	**0.81**

Table 10: Same as Table 9 but for male speakers.

	no WCCN		WCCN	
Size of hidden layer	MAE	ρ	MAE	ρ
128	**7.11**	**0.68**	7.25	0.67
256	**6.90**	**0.69**	7.06	0.69
512	6.85	0.71	**6.48**	**0.73**
1024	6.66	0.72	**6.35**	**0.74**

6.3 Neural networks ensembles

Combining results of several predictors (e.g. neural networks) can increase overall accuracy compared to one of those predictors. The common name for these types of methods are *ensemble* methods [53]. They are based on the assumption that there exists an ideal function, $f(x)$, that solving a given problem perfectly. Neural networks attempt to find a good approximation of this function, $\hat{f}(x)$. If we introduce a new *misfit* function as a deviation of approximation from

the true function, $m(x) = f(x) - \hat{f}(x)$, we can express objective of the training to be minimization of mean squared error (MSE), $\mathrm{MSE} = E[m^2]$. Having a set of n approximation functions and by assuming that their misfits are independent and have zero mean we can combine results by simple averaging from all the functions in this set and the total MSE will be $\frac{1}{n}\overline{\mathrm{MSE}}$, where $\overline{\mathrm{MSE}}$ is mean of MSEs over all functions in the set [54]. If we consider misfits as random noise, averaging all the misfits is averaging noise which means that we perform *smoothing* over the approximations on true solution. This approach is called *basic ensemble method* [54].

In this section we apply the described ensemble procedure for age estimation. To introduce diversity to our neural net ensemble,we use random initializations of networks with the exact same architecture and change the number of hidden neurons or training method.

Table 11: Combinations of different ANNs (averaged outputs).

	Male		Female	
System configuration	MAE	ρ	MAE	ρ
Base predictors				
1. $n_1 = 256$, SGD	7.06	0.69	6.11	0.77
2. $n_2 = 512$, SGD	6.53	**0.73**	5.72	0.79
3. $n_3 = 1024$, SGD	**6.35**	**0.73**	**5.49**	**0.81**
4. $n_4 = 512$, BFGS	6.66	0.71	5.69	0.79
Ensembles of base predictors				
$1 + 2 + 3$	**6.42**	**0.75**	**5.56**	**0.81**
$2 + 2 + 2$	6.45	0.73	5.63	**0.81**
$2 + 4$	6.58	0.72	5.59	0.80

Table 11 represents the results achieved with this ideology. For all the combined networks, WCCN is applied. The first four lines correspond to single networks having different numbers of hidden neurons (n_i) with jointly varied training algorithm. The last three rows show performance of a few combinations of these base networks. We selected these combinations to get various sources of diversity in one combination. The first combination address to ensemble of networks having different sizes. The second combination takes random initialization of the network weights as a source of variability. And the last ensemle is based on different training methods of networks. In each case, we simply average outputs of the individual networks and compute performance measures for the result. As expected, combination of several age estimators is helpful. The best improvement achieved for combinations of networks having the same size.

6.4 Linear discriminant analysis

In the last experiment, we investigate whether linear discriminant analysis (LDA) – as a session compensation and dimensionality reduction tool – can improve age prediction accuracy. As in the case of WCCN, we consider each speaker as a separate class. In Table 12, the results are shown for different target size for reduced input vector of 512 neurons network. The training algorithm used is SGD.

Table 12: Effect of LDA dimensionality reduction.

	Male		Female	
Target dim.	MAE	ρ	MAE	ρ
100	6.71	0.69	**5.71**	0.79
200	6.70	0.69	5.72	0.79
300	**6.46**	**0.73**	5.73	**0.80**
400 (no LDA)	6.48	**0.73**	5.72	**0.80**

In general, dimensionality reduction does not affect performance much. For the sake of speed and resource consumption, it can still be beneficial to use LDA.

6.5 Neural networks with two layers

After performing several experiments with single hidden layer neural networks, it is interesting to investigate the effect of the network architecture closer. In the current architecture, the only nonlinearity is the only hidden layer. Adding more nonlinear layers can potentially help the estimator to fit the data better. This is where inspiration for performing experiments with deep neural networks [55] came from. This section describes results of experiments performed for networks with two hidden layers.

Fixing all the other design choices as they were in the case of one hidden layer (SGD training, WCCN on), we attempt to find appropriate size for each layer of the revised network. On the first set of graphs below (Fig. 13), MAE and ρ are shown for networks having 1024 neurons in the first hidden layer and various sizes of the second layer. Fig. 14, in turn, presents the opposite case, when size of second layer was fixed to 512 neurons and the number of neurons in the first layer is varied.

It is difficult to interpret the results in Fig. 13 and Fig. 14 conclusively but one observation is that the best performance can be achieved when the sizes of two layers are equal. But even in this case, this architecture did not outperform the results achieved by a single layer network. In the best case, the performance remained the same as before. One problem could be that we used parameter values optimized for single layer network. They may require re-optimization for the

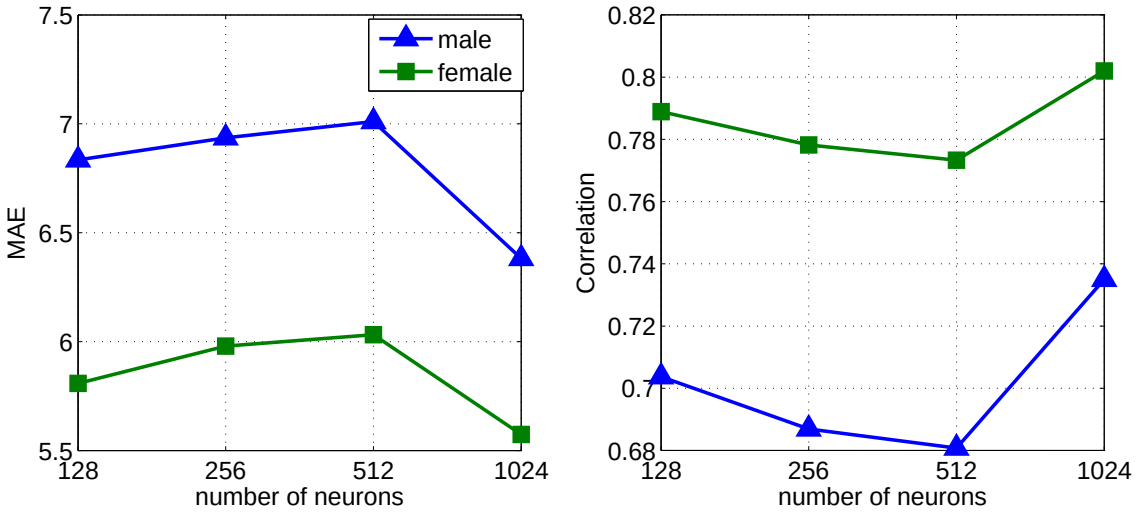

Figure 13: Performance for 2- layers network estimator of male and female speakers' age. Size of second layer is various.

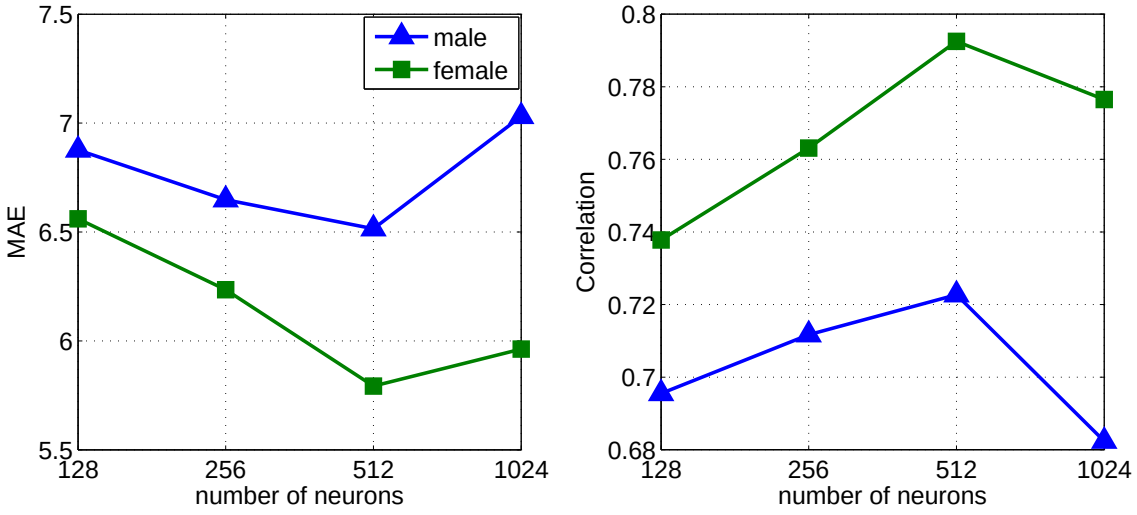

Figure 14: Performance for 2- layers network estimator of male and female speakers' age. Size of first layer is various.

2-layer structure.

Probably even more complicated architectures can provide some improvement in the age estimator performance. But due to high computational costs this direction of research was not developed further in this thesis.

7 Conclusion

This thesis was aiming at improving state-of-the-art age estimation using speech sygnals. Starting with one of the most successful methods for age estimation, utilizing i-vector representation of speech utterances, we studied a number of new systems to find out whether they could improve state-of-the-art methods in the field. Our main findings can be summarized as follows:

1. Among the different feature extraction techniques, MFCCs with short-term CMVN worked the best. Neither transformations of MFCCs nor any other features we attempted performed any better.

2. Other studied features, such as using MFCCs directly or statistical features extracted by a bottleneck neural network, were not found helpful.

3. In case of both ANN and SVR back-end, WCCN helped when treatening speakers (rather than age groups) as the classes. This agrees with the results reported for the same data in [8].

4. The second session compensation technique that we studied, LDA, did not improve age estimation accuracy.

5. Among the two compared MLP training methods, BFGS was less stable against increasing the number of neurons in the hidden layer. It was overfit with more than 256 hidden neurons while SGD was stable even with 1024 neurons. Apart from this, SGD requires less computations as it does not utilize second order derivative as opposed to BFGS.

6. A slight improvement in estimation accuracy was obtained by forming an ensemble of several networks. The most successful combination was ensemble of networks having the same size, as it was greatest improvent compared to a single network.

7. Attempts to extend network architecture by adding more hidden layers were not successful. One possible explanation is that we used for extended networks the same network parameters, optimized for single hidden layer ANN; they should be optimized again for modified architecture, which was left out for time constraints and computational reasons.

In the light of the above observations, we advise to use neural network predictor trained on i-vectors obtained from MFCC features with short-term CMVN. The recommended architecture is MLP with one hidden layer, and the training algorithm is SGD. With this setup we managed to get 4.5 % relative reduction in MAE over the baseline approach for both male and female speakers. Table 7 provides results of other age estimation approaches, mentioned in Section 2, along with result of our best system. For consistent comparison here we present results only for

Table 13: Comparison of several approaches for age estimation. Here, indexes m and f indicate results for male and female tests, respectively. Performance measures are mean absolute error (MAE) and Pearson's correlation coefficient (ρ) as detailed in Section 4.4.

			Results	
Study	Data	Methodology	MAE	ρ
Bocklet *et al.* [9]	Children of age from 5 to 11 years old	GMM-supervectors + SVR	—	ρ=0.89
Bahari *et al.* [7]	NIST 2008, 2010 SRE	GMM-supervectors + SVR	$\mathrm{MAE_f} = 7.95$, $\mathrm{MAE_m} = 7.79$	—
Bahari *et al.* [8]	NIST 2008, 2010 SRE	i-vectors + SVR	$\mathrm{MAE_f} = 5.78$, $\mathrm{MAE_m} = 6.53$	$\rho_\mathrm{f} = 0.80$, $\rho_\mathrm{m} = 0.73$
This thesis	NIST 2008, 2010 SRE	i-vectors + SVR	$\mathrm{MAE_f} = 5.75$, $\mathrm{MAE_m} = 6.65$	$\rho_\mathrm{f} = 0.80$, $\rho_\mathrm{m} = 0.73$
This thesis	NIST 2008, 2010 SRE	i-vectors + ANN	$\mathrm{MAE_f} = 5.49$, $\mathrm{MAE_m} = 6.35$	$\rho_\mathrm{f} = 0.81$, $\rho_\mathrm{m} = 0.74$

regression. The table indicates that with ANN back-end we managed not only reach but also slightly outperform state-of-the-art method.

We propose to concentrate future work on looking of alternative ways of feature extraction instead of i-vectors since back-end did not shown to have great effect on overall performance. We advise to consider closer neural networks as feature extraction technique, in this thesis we did not pay much attention to this type of features because of computational and time limits, but it may be a promising future direction of research. Architecture and parameters of the network used for feature extraction should be akso studied closer.

References

[1] S. E. Shepstone, Z.-H. Tan, and S. H. Jensen, "Audio-based age and gender identification to enhance the recommendation of tv content," *IEEE Transactions on Consumer Electronics*, vol. 59, pp. 721–729, 2013.

[2] M. Feld, F. Burkhardt, and C. Muller, "Automatic speaker age and gender recognition in the car for tailoring dialog and mobile services," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[3] M. Li, K. J. Han, and S. Narayanan, "Automatic speaker age and gender recognition using acoustic and prosodic level information fusion," *Computer Speech and Language*, vol. 27, pp. 151–167, 2013.

[4] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Tansactions on audio, speech, and language processing*, vol. 19, pp. 788–798, 2011.

[5] N. Dehak, P. A.Torres-Carrasquillo, D. Reynolds, and R. Dehak, "Language recognition via i-vectors and dimensionality reduction," in *Proceedings of Interspeech*, 2011.

[6] M. H. Bahari, R. Saeidi, H. V. hamme, and D. V. Leeuwen, "Accent recognition using i-vector, gaussian mean supervector and gaussian posterior probability supervect," in *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pp. 7344–7348, 2013.

[7] M. H. Bahari, M. McLaren, H. V. hamme, and D. van Leeuwen, "Age estimation from telephone speech using i-vectors," in *Proceedings of Interspeech*, pp. 506–509, 2012.

[8] M. H. Bahari, M. McLaren, H. V. hamme, and D. van Leeuwen, "Speaker age estimation using i-vectors," *Engineering Applications of Artificial Intelligence*, vol. 34, pp. 99–108, 2014.

[9] T. Bocklet, A. Maier, and E. Noth, "Age determination of children in preschool and primary school age with gmm-based supervectors and support vector machines/regression," in *Text, Speech and Dialogue*, vol. 5246, pp. 253–260, 2008.

[10] W. M. Campbell, D. E. Sturim, and D. A. Reynolds, "Support vector machines using gmm supervectors for speaker verifcation," *IEEE Signal Processing Letters*, vol. 13, pp. 308–311, 2006.

[11] C. M. Bishop, "Pattern recognition and machine learning," 2006.

[12] E. D. Mysak, "Pitch and duration characteristics of older males," *Journal of Speech, Language, and Hearing Research*, vol. 2, pp. 46–54, 1959.

[13] S. E. Linville, "The sound of senescence," *Journal of Voice*, vol. 10, pp. 190–200, 1996.

[14] F. Burkhardt, M. Eckert, W. Johannsen, and J. Stegmann, "A database of age and gender annotated telephone speech," in *7th International Conference on Language Resources and Evaluation (LREC 2010)*, pp. 1562–1565, 2010.

[15] N. Minematsu, M. Sekiguchi, and K. Hirose, "Automatic estimation of one's age with his/her speech based upon acoustic modeling techniques of speakers," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 137–140, 2002.

[16] C. Muller, "Automatic recognition of speakers' age and gender on the basis of empirical studies," in *Proceedings of Interspeech*, pp. 2118–2121, 2006.

[17] B. Schuller, S. Steidl, A. Batliner, F. Burkhardt, L. Devillers, C. Muller, and S. Narayanan, "The INTERSPEECH 2010 Paralinguistic Challenge," 2010.

[18] T. Bocklet, G. Stemmer, V. Zeissler, and E. Nöth, "Age and gender recognition based on multiple systems - early vs. late fusion," in *Proceedings of Interspeech*, pp. 2830–2833, 2010.

[19] P. Nguyen, D. T. Trung Le, X. Huan, and D. Sharma, "Fuzzy support vector machines for age and gender classification," in *Proceedings of Interspeech*, pp. 2806–2809, 2010.

[20] M. Kockmann, L. Burget, and J. Černocky, "Brno university of technology system for interspeech 2010 paralinguistic challenge," in *Proceedings of Interspeech*, pp. 2822–2825, 2010.

[21] G. Dobry, R. M. H. M. Avigal, and Y. Zigel, "for efficient speaker age estimation based on the acoustic speech signal," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, pp. 1975–1985, 2011.

[22] A. Martin, D. Charlet, and L. Mauuary, "Robust speech/non-speech detection using lda applied to mfcc," 2001.

[23] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Transactions on Acoustics, Speech, and Signal processing*, vol. 28, pp. 357–366, 1980.

[24] C. S. Burrus and T. W. Parks, "DFT/FFT and Convolution Algorithms: Theory and Implementation," 1991.

[25] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker verification using adapted gaussian mixture models," *Digital Signal Processing*, vol. 10, pp. 19–41, 2000.

[26] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society*, vol. 39, pp. 1–38, 1977.

[27] T. Bocklet, A. Maier, J. G. Bauer, F. Burkhardt, and E. Noth, "Age and gender recognition for telephone applications based on gmm supervectors and support vector machines," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1605–1608, 2008.

[28] X. Zhuang, J. Huang, G. Potamianos, and M. Hasegawa-Johnson, "Acoustic fall detection using gaussian mixture models and gmm supervectors," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 69–72, 2009.

[29] P. Kenny, "Joint factor analysis of speaker and session variability: Theory and algorithms," tech. rep., CRIM, 2005.

[30] "Joint Factor Analysis Matlab Demo." `http://speech.fit.vutbr.cz/en/software/joint-factor-analysis-matlab-demo`. Accessed: 15.05.2015.

[31] A. O. Hatch, S. Kajarekar, and A. Stolcke, "Within-class covariance normalization for svm-based speaker recognition," in *Proceedings of Interspeech*, 2006.

[32] S. Balakrishnama and A. Ganapathiraju, "Linear discriminant analysis - a brief tutorial," *Advances in Neural Information Processing Systems*, vol. 13, pp. 402–408, 2001.

[33] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[34] A. J. Smola and B. Scholkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, p. 199–222, 2004.

[35] "LIBSVM – A Library for Support Vector Machines." `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`. Accessed: 15.05.2015.

[36] "SVMTorch." `http://bengio.abracadoudou.com/SVMTorch.html`. Accessed: 15.05.2015.

[37] "LS-SVMlab." `http://www.esat.kuleuven.be/sista/lssvmlab/`. Accessed: 15.05.2015.

[38] J. A. K. Suykens, T. V. Gestel, J. D. Brabanter, B. D. Moor, and J. Vandewalle, *Least Squares Support Vector Machines*. World Scientific, 2002.

[39] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[40] L. Bottou, "Online algorithms and stochastic approximations," in *Online Learning and Neural Networks*, 1998.

[41] J. J. E. Dennis and J. J. Moré, "Quasi-newton methods, motivation and theory," *SIAM Review*, vol. 19, pp. 46–89, 1977.

[42] C. G. Broyden, "Quasi-newton methods and their application to function minimisation," *Mathematics of Computation*, vol. 21, pp. 368–381, 1967.

[43] J. Nocedal, "Updating quasi-newton matrices with limited storage," *Mathematics of Computation*, vol. 35, pp. 773–782, 1980.

[44] R. Caruana, S. Lawrence, and L. Giles, "Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping," *Advances in Neural Information Processing Systems*, vol. 13, pp. 402–408, 2001.

[45] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[46] A. Krogh and J. Hertz, "A simple weight decay can improve generalization," *Advances in Neural Information Processing Systems*, vol. 4, pp. 950–957, 1992.

[47] O. Viikki and K. Laurila, "Cepstral domain segmental feature vector normalization for noise robust speech recognition," *Speech Communication*, vol. 25, pp. 133 –147, 1998.

[48] S. Cumani, O. Glembek, N. Brummer, E. de Villiers, and P. Laface, "Gender independent discriminative speaker recognition in i-vector space," in *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pp. 4361–4364, 2012.

[49] P. A. Torres-Carrasquillo, E. Singer, M. A. Kohler, and R. J. Greene, "Approaches to language identification using gaussian mixture models and shifted delta cepstral features," in *Proceedings of Interspeech*, 2002.

[50] V. Mitra, M. McLaren, H. Franco, M. Graciarena, and N. Scheffer, "Modulation features for noise robust speaker identification," in *Proceedings of Interspeech*, 2013.

[51] M. Slaney, "An Efficient Implementation of the Patterson-Holdsworth Auditory Filter Bank," tech. rep., Perception Group—Advanced Technology Group, Apple Computer, Inc, 1993.

[52] T. N. Sainath, B. Kingsbury, and B. Ramabhadran, "Auto-encoder bottleneck features using deep belief networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4153–4156, 2012.

[53] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, pp. 993–1001, 1990.

[54] M. P. Perrone and L. N. Cooper, "When networks disagree: Ensemble methods for hybrid neural networks," *Neural networks for speech and image processing*, 1992.

[55] Y. Bengio, "Learning Deep Architectures for AI," 2009.